An approximation algorithm for matroid covering (Theoretical Computer Science and its Applications)

AUTHOR(S):

Kawano, Shinichiro; Otachi, Yota; Yamazaki, Koichi

数理解析研究所講究録 1426 巻 2005 年 166-171

An approximation algorithm for matroid covering

川野 晋一郎 (Shinichiro Kawano) 大舘 陽太 (Yota Otachi) 山崎 浩一 (Koichi Yamazaki) *

Department of Computer Science, Faculty of Engineering, Gunma University
群馬大学・工学部・情報工学科

1 Introduction

In this paper we address the following matroid covering problem: Given a matroid $M = (S, \mathcal{I})$ (S is the *ground set* and $\mathcal{I}$ is the collection of *independent set*), find to minimum number of independent sets which cover S. We call the minimum number *covering number* of M. For example, let G be a graph and M be the graphic matroid of G. Then the covering number of M is edge-arboricity of G (the edge-arboricity of a graph is the minimum number of forests which cover the edge set of the graph). Matroid covering problem has many applications (see [3] and also Section 5).

It is known that the covering number can be computed in polynomial time, however, the time complexity is high. For instance the edge-arboricity of a graph can be computed in $O(mn \log n)$ [3], where n and m denote the number of the vertices and the edges, respectively. Our goal in this paper is to find simple and fast approximation algorithms with small approximation ratios for the matroid covering problem.

2 Preliminaries

Let $M = (S, \mathcal{I})$ be a matroid. The *ground set* S and the collection $\mathcal{I}$ of *independent sets* are denoted by $S(M)$ and $\mathcal{I}(M)$ respectively. The rank function of M is denoted by ρ. For a subset T of S, $M' = (T, \mathcal{I}')$ is also a matroid where $\mathcal{I}' = \{X : X \subseteq T, X \in \mathcal{I}\}$. The matroid M' is called the *restriction* of M to T and we denote it by $M \mid T$. A *flat* F of M is a subset of S such that $\rho(F \cup \{x\}) > \rho(F)$ for any $x \in S - F$. A *hyperplane* is a flat of rank $r(M) - 1$.

3 An approximation algorithm for matroid covering

In this section we present approximation algorithms for matroid covering. For some matroid classes, the algorithm guarantees the solution to be at most c times the optimal for some constant c (which depends on the class, see section 5).

3.1 A general algorithm

In this subsection, for the matroid covering problem, we describe a general algorithm which is designed within a general framework as follows.

Algorithm GA (General Algorithm)
input: a matroid $M = (S, \mathcal{I})$

*This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (C), 16500008, 2004.

output: an independent partition

```
1:  M_1 := M;  S_1 := S;  I_1 := I;  i := 1;
2:  while S_i ≠ ∅ do
3:  begin
4:     find a flat S_{i+1} of M_i;  DE_i := S_i − S_{i+1};
5:     M_{i+1} := M_i | S_{i+1};  i := i + 1;
6:  end;
7:  cnt := i;  j := 0;
8:  while ∃h such that DE_h ≠ ∅
9:  begin
10:    j := j + 1; P_j := ∅;
11:    for i := 1 to cnt do
12:       if DE_i ≠ ∅ then
13:          begin
14:             choose any x ∈ DE_i;
15:             DE_i := DE_i − {x};
16:             P_j := P_j ∪ {x};
17:          end
18: end;
19: output P_k (1 ≤ k ≤ j);
```

Note that the value of j at the end of the algorithm equals $\max\{|DE_1|, \ldots, |DE_{cnt}|\}$, we will refer to it as *size of output*. Theorem 3.1 shows the correctness of the algorithm GA.

Theorem 3.1 *Let $M = (S, \mathcal{I})$ be a matroid, F be a flat of M, and A is an independent subset of F. Then $A \cup \{x\} \in \mathcal{I}$ for any $x \in S - F$.*

Proof. Since F is a flat, $\rho(F \cup \{x\}) > \rho(F)$ for any $x \in S - F$. If $A \cup \{x\} \notin \mathcal{I}$ then $\rho(A \cup \{x\}) = \rho(A)$. Thus we have $\rho(F \cup \{x\}) = \rho\Big((A \cup \{x\}) \cup F\Big) \leq \rho(A \cup \{x\}) + \rho(F) - \rho(A) = \rho(F)$. But this is a contradiction. □

3.2 A handy algorithm

In this subsection, we introduce a handier version of the algorithm GA for the matroid covering problem. The version takes as input the flat matrix which is defined as follows.

Let A be a matrix and R be a row in A. $SM(A, R)$ denotes the submatrix of A obtained by deleting the columns C such that the entry in R and C is nonzero. A *flat matrix* of a matroid $M = (S, \mathcal{I})$ is defined recursively as follows:

1. The empty 0-by-0 matrix is a flat matrix.
2. If
   (a) there is a mapping f such that for any row R in A f assigns $SM(A, R)$ to a flat of M,
   (b) for any row R in A, $SM(A, R)$ is a flat matrix, and
   (c) the rank of the $f(SM(A, R))$ is at most the number of nonzero row vectors in $SM(A, R)$,

   then A is also a flat matrix.

Now we are ready to describe the handier version. Using the flat matrix, the algorithm GA can be modified for more practical use as follows.

Algorithm FMA (Flat Matrix Algorithm)
input: a flat matrix A of a matroid $M = (S, \mathcal{I})$.
output: an independent partition
1: $A_1 := A$;
2: **while** A_i has a nonzero row **do**
3: **begin**
4: find a minimum weight
nonzero row vector MR_i of A_i;
5: $A_{i+1} := SM(A_i, MR_i)$;
6: $DE_i := \{e \in S \mid e$ corresponds to a column
which is in A_i but not in $A_{i+1}\}$;
7: **end**;
8: $cnt := i$; $j := 0$;
9: **while** $\exists h$ such that $DE_h \neq \emptyset$
10: **begin**
11: $j := j + 1$; $P_j := \emptyset$;
12: **for** $i := 1$ **to** cnt **do**
13: **if** $DE_i \neq \emptyset$ **then**
14: **begin**
15: choose any $x \in DE_i$;
16: $DE_i := DE_i - \{x\}$;
17: $P_j := P_j \cup \{x\}$;
18: **end**
19: **end**;
20: **output** P_k $(1 \leq k \leq j)$;

Theorem 3.2 *Let A be a flat matrix of a matroid $M = (S, \mathcal{I})$ such that each column of A has at most k nonzero-entries. Then the size of output of the algorithm is at most k times the covering number of M.*

Proof. Let i be an index such that $|DE_i| = \max_j |DE_j|$ and let us denote $\{e \in S \mid e$ corresponds to a column in $A_i\}$ by F_i.

Since $k|F_i| \geq$ (the number of nonzero-entries in A_i) $\geq$ (the number of nonzero rows in $A_i)|DE_i| \geq \rho(F_i)|DE_i|$, we have $|F_i|/\rho(F_i) \geq |DE_i|/k =$ (the size of output)$/k$. Now the theorem follows Edmonds' covering theorem that for a matroid $M = (S, \mathcal{I})$, the covering number of M equals $\max_{X \subseteq S} \lceil |X|/\rho(X) \rceil$. □

4 Straight greedy algorithm

A greedy approach is one of the most simple and faster ways to obtain an approximate solution. In this section we estimate the performance ratio of the following *straight* greedy algorithm.

Algorithm straight greedy algorithm
input: a matroid $M = (S, \mathcal{I})$
output: an upper bound of the covering number of M
1: $m := 0$;

2: **while** $S(M) \neq \emptyset$ **do**
3: **begin**
4: find a maximum independent set T of M;
5: $M := M|(S(M) - T)$
(i.e. the restriction of M to $S(M) - T$);
6: $m := m + 1$;
7: **end**;
8: **output** m;

How good is the straight greedy algorithms? What is the performance ratio of the straight greedy algorithm? Let us analyze the worst case behavior of the straight greedy algorithm for the edge-arboricity problem. Consider the edge-arboricity of the graph in Figure 1. The edge-arboricity of the graph is 2 as shown in Figure 2, but the straight greedy algorithm may output 4 as illustrated in Figure 3.

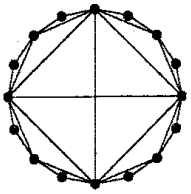

Figure 1: A graph for which the straight greedy algorithm may yield a bad solution.

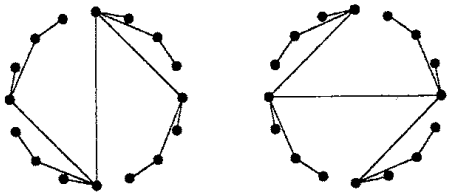

Figure 2: An optimal partitioning.

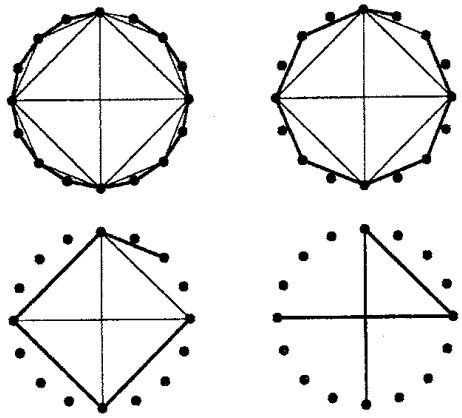

Figure 3: A worst case behavior.

From the above observation, it is not difficult to see that we can make a graph G with 2^n vertices such that the edge-arboricity of G is 2 and the straight greedy algorithm may produce n partitions. As we will see (Section 5), the performance ratio of algorithm FMA for the problem is 2.

5 Applications

5.1 Graphic matroid

It is known that the covering problem on a graphic matroid corresponds to the problem computing the edge-arboricity of a graph. Let us adapt the algorithm FMA to the edge-arboricity

problem. H is a hyperplane iff $S - H$ is a *cocircuit*. In a graphic matroid, a cocircuit is a minimal cutset. Thus, we have the following algorithm from the algorithm FMA.

Algorithm ARBOR-MINCUT
input: a graph $G = (V, E)$
output: an upper bound of the edge-arboricity of G
1: find a minimal cutset C of G;
2: compute the connected components
G_1 and G_2 of $G - C$;
3: /* Note that $G - C$ has two components
because C is minimal. */
4: ARBOR-MINCUT :=
$\max\{|C|,$ ARBOR-MINCUT(G_1),
ARBOR-MINCUT$(G_2)\}$;

One of cheapest way to find a cutset (not necessary minimal) is to search for the edges incident to a vertex of minimum degree (because such a set of edges is a disjoint union of cutsets). The algorithm FMA algorithm derives the following algorithm, which is the same as the algorithm proposed in [1]. The algorithm can be easily implemented and its running time is $O(|V(G)| + |E(G)|)$.

Algorithm ARBOR-MINDEG
input: a graph $G = (V, E)$
output: an upper bound of the edge-arboricity of G
1: $m := 0$;
2: **while** $E(G) \neq \emptyset$ **do**
3: **begin**
4: find a vertex v with minimum degree of G;
5: **if** $d_G(v) > m$ **then** $m := d_G(v)$;
6: $G := G[V(G) - \{v\}]$;
7: **end**;
8: **output** m;

Take the incident matrix of G for the flat matrix, it is then clear that the approximation ratio is 2. By a simple observation, we have the following result.

Lemma 5.1 *If the edge-arboricity of a graph G is at most k, then any subgraph H of G has a vertex of degree at most $2k - 1$.*

Proof. From Edmonds' covering theorem, if the edge-arboricity of a graph G is at most k, then $\lceil \frac{|E(H)|}{V(H)-1} \rceil \leq k$ holds for any subgraph H of G. So for any subgraph H of G, the average degree of H is less than $2k$ (note that $2\frac{|E(H)|}{V(H)} < 2\lceil \frac{|E(H)|}{V(H)-1} \rceil \leq 2k$), which completes the lemma. □

Corollary 5.2 *For a planar graph G, if* ARBOR-MINDEG *outputs 4 or 5 for G then the edge-arboricity of G is 3.*

Proof. It is well-known that the edge-arboricity of any planar graph is at most 3. If ARBOR-MINDEG outputs 5 for G, then the edge-arboricity of G is at least $\lceil 5/2 \rceil = 3$. If ARBOR-MINDEG outputs 4 for G, then there exists a subgraph which has no vertices of degree less than 4. From the above lemma, the edge-arboricity of the input graph is at least 3. □

5.2 Transversal matroid

Let us consider the following scheduling problem. There are jobs and machines. Each job can be processed on at least one machine, and processed in a unit time (on any available machine). Each round has a unit time, so each machine processes at most one job in each round. Then problem is to find the minimum number of rounds needed to process the all jobs. This scheduling problem can be viewed as a covering problem on a transversal matroid.

Let us consider the example shown in Table 1. A straight greedy strategy (i.e. taking a maximum matching in each round) might output "3 rounds" (See Figure 4). However, as shown Figure 5, the minimum number of rounds is 2, and algorithm FMA can achieve the optimal rounds in the example. If for any job J_j the number of machines which can process J_j is at most k, then algorithm FMA for the problem has performance ratio of at most k (for the example, k is 2).

	J_1	J_2	J_3	J_4	J_5	J_6	J_7
M_1	1	0	0	0	1	1	0
M_2	1	1	0	0	0	0	0
M_3	0	1	1	0	0	1	1
M_4	0	0	1	1	0	0	0

Table 1: The relationships between jobs and machines.

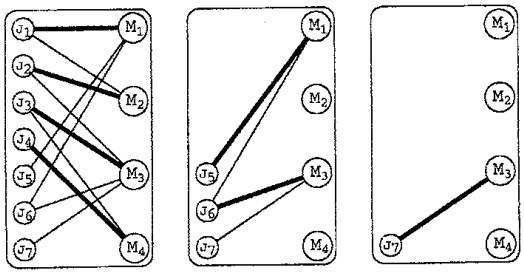

Figure 4: A bad scheduling.

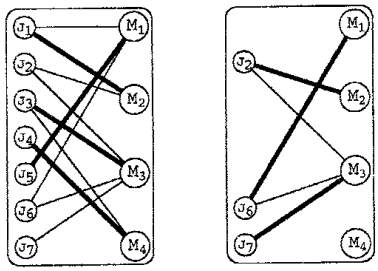

Figure 5: An optimal scheduling.

References

[1] S.Arikati, A.Maheshwari and C.Zaroliagis, Efficient Computation of Implicit Representations of Sparse Graphs, Discrete Applied Mathematics Vol. 78 (1997) 1-16.

[2] J.Edmonds, Lehman's switching game and a theorem of Tutte and Nash-Williams, J. Res. Nat. Bur. Standards Sect. B 69B (1965) 73-77.

[3] H.N.Gabow and H.H.Westermann, Forests, Frames, and Games: Algorithms for Matroid Sums and Applications, Algorithmica Vol.7 (1992) 465-497.